8510t — Integrating the AT&T 8510T ISDN Telephone with a UNIX Workstation

Kelley Hu (khu@nlm.nih.gov)
R. P. Channing Rodgers (rodgers@nlm.nih.gov)
ENRG Group, Computer Science Branch
Lister Hill National Center for Biomedical Communications
U.S. National Library of Medicine

ENRG Technical Report 99-1 | October 1999 | 8510t Version 1.1

Abstract

The *8510t* application is a Bourne shell script which invokes one of two associated Java applications to control an AT&T 8510T ISDN telephone, which is connected to the serial port of a Sun workstation. It can place two-party and three-party (conference) calls, and can also be used to trigger telephone dialing in conjunction with a properly configured World Wide Web client. With appropriate additional work, it should be capable of operating on PCs operating under Windows™ and Apple Macintosh™ computers as well.

8510t — Integrating the AT&T 8510T ISDN Telephone with a UNIX Workstation

Kelley Hu (khu@nlm.nih.gov)
R. P. Channing Rodgers (rodgers@nlm.nih.gov)
ENRG Group, Computer Science Branch
Lister Hill National Center for Biomedical Communications
U.S. National Library of Medicine

ENRG Technical Report 99-1 October 1999 8510t Version 1.1

1. Introduction

The *8510t* application is a Bourne shell script which invokes one of two Java applications to control an AT&T 8510T ISDN telephone which is connected to the serial port of a Sun workstation. It can place either two-party or three-party (conference) calls. You can run this application on the command line with either one or two phone numbers as its arguments; the ISDN telephone that is connected to the computer will start dialing the other telephone(s). A **–h** option allows the ISDN telephone to be put on hold several seconds after a 3-party call is established, allowing the user to employ his phone when away from the desk to make calls, without disturbing others. A **–d** option allows the initiation of the calls to be delayed, in the event that the user is starting the application via modem over a phone line which he wishes the application to call, allowing time to free up the line.

This package also supports the use of the ISDN telephone in conjunction with a World Wide Web client. Hypertext anchors can be embedded in a document, such that when the anchor is selected by the user, the phone is made to dial a particular phone number. This is most obviously useful in the context of a phone directory service, and it is used in this manner within the U.S. National Library of Medicine's Lister Hill National Center for Biomedical Communications, where the application was written.

2. Requirements

1) UNIX workstation running JDK 1.1 or JDK 1.2.
2) AT&T 8510T ISDN telephone, connected to workstation with a cable that connects pins 1-8, 20, and 22, straight through.
3) Secure shell (*ssh*): required *only* if you wish to support the "multihost dialout" mechanism described below.
4) A World Wide Web client running on the workstation, which is capable of being configured to support a new file type: required *only* if the *8510t* program is to be used to dial numbers by selecting an appropriately specified anchor in a Web document.
5) Access allowing the installation of a CGI application on a web server: required *only* if the server in question is to be able to offer documents which trigger client-side phone dialing.

This package has been tested only on the Sun UltraSPARC 60 under Solaris 2.6 and JDK 1.1 or JDK 1.2. Please note the other issues appearing in section 6.

3. How It Is Used

A few definitions are in order. There are three telephone numbers at issue:

1) *ISDN telephone number*: the telephone number corresponding to the ISDN telephone that is connected to the computer that is executing *8510t*
2) *phone number 1*: the telephone number for the first party to be connected to
3) *phone number 2*: the telephone number for the second party to be connected to

By definition, a true "three-way conference call" requires that all three of the above numbers be unique. If the *ISDN telephone number* is identical to either *phone number 1* or *phone number 2*, the phone is actually being used for a traditional two-party call, and the *8510t* script will invoke a Java application that is different from that used when placing a three-party conference call.

A call is initiated from the UNIX command line invoking the shell script *8510t* as follows:

`[...]/8510t [-d sec] [-h] [-u URL] <`*phone number 1*`> [<`*phone number 2*`>]`

You must supply, at a minimum, *phone number 1*. If only one phone number is supplied, a two-party call is established, and if two numbers are provided, a three-party conference call is established. If one of the two numbers is identical to the ISDN telephone number, a two-party call is established.

The script translates alphabetic characters appearing in phone numbers to their corresponding integers, and strips out non-alphanumeric characters (such as whitespace or parentheses or dashes) and sends the resulting number to a Java application which actually causes the phone to dial the number. For example, the dial string "1 (800) DIA-LATT" would be translated to "18003425288".

In the event that a three-party call is being executed and one of the call presences of the ISDN telephone is in use, *8510t* employs a "multihost dialout" mechanism, in which it tries to go down a (user-supplied) list of other hosts on which *8510t* has been installed, and place the call on the ISDN sets attached to those computers, continuing down the list of such hosts until a call is successfully completed or the list of hosts is exhausted. The list is supplied in the file *$(HOME)/.8510t*, which contains lines of the form:

`<hostname> <login>`

where `<hostname>` is the name of another computer on which *8510t* can be run by the user, and `<login>` is the user's login name on that host (this format is identical to that used by *.rhosts* and *.shosts* files). NOTE: we assume that secure shell, *ssh*, is being used, and that entries of the above form also appear in the user's *.shosts* file, so that the application will not be prompted for a password. A "time-to-live" mechanism is implemented to prevent infinite loops. The **–h** option is always automatically enabled for multihost dialout.

The **–d** option implements a delay of the specified number of seconds before placing the call(s); this is useful if the script is being activated remotely by a modem connection over one of the phone lines that is to participate in a three-party conference call, allowing the user time to log off and free the line so that it will not be busy when the application dials it.

The application may also be used in conjunction with the World Wide Web. This involves three requirements:

1) The installation on a web server of the supplied CGI application application, *phone.cgi*.

2) The inclusion with hypertext document on the server of phone number anchors of the form:

```
http://<hostname>/cgi-bin/phone.cgi?num=913011234567
```

3) The configuration of the user's Web client to trigger the *8510t* script in response to receipt of a document of type *x-8510t* from a Web server.

When the user clicks on a phone number anchor, a request is sent to the server, which executes *phone.cgi*, which sends back a small file to the client, of document type `x-8510t`. The client triggers the local copy of the *8510t* script, passing the original URL to the script as the argument following a **–u** option. The script picks out the phone number from the URL and processes it as it does all phone numbers.

Finally, the system administrator may exercise control over use of the phone application by editing a configuration file. The file may contain lines beginning with a "#" character (ignored as comment lines) or lines of the form:

`<hostname>:<number_1> [number_2] .... [number_n]:[login_1] [login_2] ...`

where `<hostname>` is the name of a workstation on which *8510t* is to be allowed to operate, and `<number_1> [number_2] .... [number_n]` are the various digit patterns which are to be recognized as belonging to the ISDN telephone attached to that workstation, and `[login_1] [login_2] ...` are the login names of the users who are authorized to use the application on that host (if this field is empty, all users are authorized). For example:

```
baggins:913011234567 1234567 34567:smith jones doe
```

specifies that:

1) The application is allowed to run on host *baggins*;

2) The three specified digit strings should be recognized as valid representations of the phone number of the ISDN telephone attached to the serial port of baggins;

3) Only the users with login names "smith," "jones," and "doe" are allowed to use the application.

If no configuration file is found by the script, it allows the application to be run on any host by any user, and will fail if one of two numbers supplied to the script corresponds to that of the ISDN set.

4. Installation

This package has been tested only on the Sun UltraSPARC 60 under Solaris 2.6 with JDK 1.1.6 and Java2SDK (JDK 1.2_04). The following instructions apply only to that platform. We welcome contributions from others with respect to installation on other platforms.

4.1. Installation of Telephone

1) On each host on which the application is to be run, be certain that an AT&T ISDN 8510T phone telephone is connected to the workstation's serial port "B" using a properly wired cable (see above).

2) Place the ISDN telephone in AT mode. The following instructions refer to the eight buttons at the top of the AT&T 8510T telephone, just below the LED display. Note that the LED display changes during this process, so that the display describes the function of the topmost row of buttons, referred to here as "softkeys." Press the following sequence of buttons:

   Menu
   Next
   Data (softkey)
   DataOptions (softkey)
   Change (softkey)

   At this point, the "`SET DATA MODE`" display will appear. Press the following sequence of buttons:

   Skip (softkey)
   Skip (softkey)
   Skip (softkey)

   At this point, the "`SET LOCAL MODE`" display will appear. Press the following sequence of buttons. Toggle using the "+" (softkey) and "-" (softkey) buttons until the display reads "`SET LOCAL MODE: AT`" (the possible settings are: `AT`, `EnhAT`, `CMD`, and `OFF`). Finally, push the "Save" (softkey) (the unit may beep at this point). Press the "Menu" button twice to return to the (default) display of date and time.

3) Call progress monitoring controls which sounds are heard from the telephone when it is acting under computer control; these sounds include dial tone and touch-tone (DTMF, or Dual-Tone Multifrequency Signals) sounds. To set the status of call progress monitoring, press the following sequence of buttons:

   Menu
   Option (softkey)
   Next
   Next
   Next
   Call-Progress (softkey)

   At this point, th "`PC CALL PROGRESS`" display will appear. This may be set to one of the three following modes by selecting the appropriate softkey:

   Off When the handset is on hook, the speaker on the telephone will *not* play the dial tone and touch-tone sounds for any call initiated by the computer.

   Delayed
   When the handset is on hook, the speaker on the telephone will *not* play the dial tone and touch-tone sounds for any call initiated by the computer, but *will* play the sounds occurring after the last digit is dialed, including ring-back and busy tones.

   On When the handset is on hook, the speaker on the telephone will play the dial tone and the touch-tone sounds for any call initiated by the computer.

Once the appropriate state has been selected, push "Done" (softkey). A message appears in the LED display at this point, describing how the telephone will behave within the selected state. Pushing buttons other than the "Done" softkey during this message may cause the telephone to beep (and ignore the key press). Once the display returns with the "Done" softkey at the right of the display, press "Done" once more, then press "Menu" to return to the (default) display of date and time.

The setting you select here depends upon how you intend to use the **8510t** application. For example, if a telephone is being shared by a user who sits by it, as well as a remote user who employs it via **8510t** (most often to call the person sitting by the telephone), the *delayed* setting might be optimal. If **8510t** is used on that telephone only by the person sitting by the phone, the *on* setting might be best. If the telephone is sitting in a locked closet and is used only remotely by a user employing the *off* setting might be optimal. Note that whether or not anything is heard *after* a call is established is determined by use of the **–h** option.

4.2. Initialization of Serial Port

1) You may require the assistance of your local system administrator at this point if you do not have administrative (root) access to the computer. As root, on each host on which the application is to be run, run the *admintool* application, and edit the settings for serial port B by selecting "Browse -> Serial Ports" from the menu bar at the top of the tool, clicking on the line for Port b in the main window, and then selecting "Edit -> Modify" from the menu bar at the top of the tool. A window labeled "Admintool: Modify Serial Port" will open. Within that window, employ the following settings:

   Template: Terminal - Hardware
   Detail: Basic
   Port: b
   Check the "Service Enable" box
   Baud Rate: 9600
   Terminal Type: tvi925

   Then click the "OK" button.

   Note: Selecting the "Expert" button for detail expands the window with additional settings. Although we believe that expert settings are not needed, we include them here for completeness:

   Options: Check "Initialize Only" and "Software Carrier"
   Login Prompt: "ttyb login:"
   Comment: (leave blank)
   Service Tag: ttyb
   Port Monitor tag: zsmon
   Expert Options: Check "Create utmp Empty"
   Service: "/usr/bin/login"
   Streams Modules: "ldterm,ttcompat"
   Timeout (secs): "Never"

4.3. Installation of Telephone Communications Software

1) On each host on which the application is to be run, be certain that the following (native threads) version of the Java Developer's Kit (JDK) is installed:

Operating System:	Solaris Version 2.x Running on the SPARC architecture
Java VM Version:	1.1 or 1.2
Java Vendor:	Sun Microsystems Inc.
Java Vendor URL:	`http://www.sun.com/`

   JDK 1.1.6 is installed in */usr/java1.1* by default; JDK 1.2 is installed in */usr/java1.2* by default.

2) **For Java 1.1**: As root, on each host on which the application is to be run, copy the file *classes/javax.comm.properties* to <*JDK*>*/lib* or <*JRE*>*/lib* directory, where <*JDK*> is the top-level directory for the local Java Development Environment (*/usr/java1.1* for JDK 1.1.6 under Solaris 2.6).
   For Java 1.2: As root, on each host on which the application is to be run, copy the file *classes/javax.comm.properties* to <*JDK*>*/jre/lib*, where <*JDK*> is the top-level directory for the local Java

Development Environment (*/usr/java1.2* for JDK 1.2 under Solaris 2.6).
THIS IS VERY IMPORTANT: if this is not done, the serial port can not be found by the *8510t* application.

3) Edit the file named *Config*, in the top level source directory, setting the values of the variables found there so as to be appropriate for your site. These include variables which control the installation locations for the components of the package as well as various UNIX commands used by them.

   A number of variables are used to generate the POSTSCRIPT and PDF versions of the manual, as well as the ASCII README file, all of which accompany the *8510t* package. You need not worry about them, and should not trigger the *make* targets that produce these documents:

   BIB
   BIBINDEX
   PRINTER
   PS2PDF
   PSROFF

4) Install the client package:

   ```
   make install
   ```

5) Edit the configuration file governing which hosts and users are to be allowed to use the application. It is strongly advisable to do this, otherwise the application can be used by any user on any system from which the application is accessible.

   Lines in the configuration file that begin with "#" are considered comments and ignored. All other lines must be of the form:

   <*hostname*>:<*number_1*> <*number_2*> .... <*number_n*>:<*login_1*> <*login_2*> ...

   where <*hostname*> is the name of a workstation on which *8510t* is to be allowed to operate, and <*number_1*> <*number_2*> .... <*number_n*> are the various digit patterns which are to be recognized as belonging to the ISDN telephone attached to that workstation, and <*login_1*> <*login_2*> ... are the login names of the users who are authorized to use the application on that host (if this field is empty, all users are authorized). This facilitates some control over usage.

6) Optionally, for each user on each host on which the application is to be run by that user, create a file *$HOME/.8510t*, containing the following lines:

   <*host name 1*> <*user name 1*>
   <*host name 2*> <*user name 2*>
   [...]
   <*host name n*> <*user name n*>

   where <*host name #*> is the name of a UNIX host on which the user has an account and is able to run the *8510t* application; and <*user name n*> is the user's login name on that host. Every line appearing in this file *must* also appear in the file *$HOME/.shosts*, to allow *ssh* to operate without inquiring for a password. This file enables the "multihost dialout" feature of *8510t*: if the ISDN telephone is busy, the script tries to run *8510t* on each of the hosts appearing in the file *.8510t*, stopping at the first host on which the conference call is successfully placed. The *.8510t* file must be owned by the user, and must not have write permissions for anyone else.

4.4. Test the Software

1) Ensuring that the path to the application appears in your PATH environment variable, test three-party conference dialing as follows:

   `8510t <phone number 1> <phone number 2>`

   For example, suppose that *phone number 1* is 301-123-4567, that the number is a long distance call, that dialing a "9" is required to obtain an outside line, and that dialing a "1" is required to initiate a long distance call. The *8510t* command would look like this:

```
./8510t 913011234567 91800CALLATT
```

resulting in the ISDN telephone connecting *phone number 1* first; once that phone is taken off hook, the ISDN telephone connects *phone number 2* (91800CALLATT). If the ISDN telephone is busy, and the user has a *.8510t* file, the multihost dialout mechanism described above will be invoked, and if either *phone number 1* or *phone number 2* is identical to the ISDN telephone number, as specified in the *8510t* configuration file, the script will actually place a two-party call instead of a three-party conference call.

Test two-party dialing with a command of the form:

`8510t <`*phone number 1*`>`

If web-based support for the *8510t* script has been installed, test it by displaying (within a Web client) a document with an anchor of the appropriate type, and select that anchor. An example of such a document appears in the file *server/html/phone.html*.

4.5. Installation of World Wide Web Support

4.5.1. Server-Side Installation

An administrator wishing to enable a Web server to offer URLs that trigger phone dialing must install the *perl* application *phone.cgi* (located in *server/cgi-bin*) as a CGI application on the server. On most Apache server installations, this involves copying the file into the Apache *cgi-bin* directory, assigning the user and group affiliations to "root" (or whatever owner/group you employ for web services at your site), and setting permissions to 755 (`-rwxr-xr-x`). The *perl* path in the *phone.cgi* script should be updated to reflect the location of *perl* on the server system.

Once this is done, you can create documents, statically or dynamically, containing anchors that will trigger phone dialing on a client on which the *8510t* application has been installed. For examples, see the file:

server/html/phone.html

which provides both examples of anchors, and the use of a form, which allows the user to manually specify the number to be dialed.

4.5.2. Client-Side Installation

The World Wide Web client that is being used to trigger phone dialing must be configured so as to trigger the *8510t* application correctly when it receives a document of type *8510t*. This can generally be accomplished by adding a line to the file *$HOME/.mime.types* of this form:

```
application/x-8510t
```

and an entry in the file *$HOME/.mailcap* of this form:

```
application/x-8510t;/depot/bin/8510t -u %u
```

Web clients often provide a graphical user interface which allows such external applications to be enabled without directly editing the aforementioned files. For example, under Netscape Navigator on X Windows, Selecting "Edit," then "Preferences," then "Navigator" and finally "Applications," starts a pop-up listing various helper applications. Selecting "New..." (or selecting a pre-existing definition for application/x-8510t, if such exists, followed by "Edit") opens a window in which one would enter

```
application/x-8510t
```

into the text box labeled "MIMEType:", then check the button labeled "Application:" and enter

```
/depot/bin/8510t -u %u
```

into the text box to the right of this button. Then selecting "OK" writes out the new settings and causes the pop-up to vanish.

5. Troubleshooting

1) If this application is interrupted, as with a control-C, it is possible that on occasion the Java process might not die, and would hence remain in control of the serial port to which the telephone is connected.

   To fix this, use the UNIX command:

   ```
   ps -aef | grep -i java"
   ```

   to determine the process ID (pid) for the *8510t* Java process, then issue the following command:

   ```
   kill -9 <pid>
   ```

   where <*pid*> is the process ID number for the *8510t* Java process. In principle, the *8510t* Bourne shell script should kill the Java process for you if it is interrupted, so this condition should rarely if ever arise.

2) If the application is started, issues a command to the telephone, and then is killed, the telephone will continue to act on the last command send to it. If this is a dial command, this may take some time to execute, and if the *8510t* application is restarted in the mean time, an error message of the following sort may appear:

   ```
   ISDN.SerialConnectionException: Port currently owned by unknown
   Solaris Application.
   ```

   If this occurs, wait for a period of time and retry the command.

3) If you obtain no response from the ISDN set, check the following:

   Is the serial cable correctly pinned?
   Is the ISDN telephone correctly connected?
   Are the serial port settings as described above?
   Is the JDK installed?
   Is the CLASSPATH variable set correctly?
   Is the ISDN telephone in AT mode? (see paragraph 2 in section 4.1)

6. Remaining Problems & Limitations

In principle, it should be possible to get this system working under other flavors of UNIX and under non-UNIX operating systems such as Windows and MacOS. At present, however, JavaSoft's Java Communications API 2.0, which this package employs for serial port communications, only supports Solaris and Windows 95/98/NT. Information about the availability of serial port drivers for Linux is available (at the time of this writing) from the JavaSoft web site (`http://java.sun.com/`).

Certain limitations of the 8510T telephone make it less than ideal as a shared device. For example, if a call presence is in use by a person sitting at the phone, and a remote user triggers an outgoing call on the second call presence, the speakers using the first call presence will hear the DTMF dial-tone sounds as the second user's number is being dialed, which is disruptive.

With certain handsets we have observed that use of the application through a web client interface is only erratically successful (though manual use is consistently successful).

7. Possible Future Features (Things To Do)

There are a number of features we would like to see added to this system, and buglets that we would like to fix. Please contact the authors if you address any of these issues:

1) The system currently uses work-arounds for certain aspects of the 8510T, such as having to wait several seconds to place the unit on hold for the **–h** option (the command failed if we issued it immediately). See point 5 in section 8.

2) Devise a daemon-based service that allows the system to be controlled by key sequences pressed by a caller dialing in on the phone line.

3) Port the script to Windows (the Java components should already work, but a substitute for the *8510t* script would have to be devised). Perl and Java are good candidates for writing a "universal" script replacement. A port to MacOS awaits support for Mac serial interfaces in the Java Communications API.

4) Add a redial option to cause the application to try again at a specified interval if it gets a busy signal from one of the remote phones being called, and to retry conference calling if the local telephone is busy.

5) Address potential nuisance and security issues. For example, a nefarious party could set up a server, and somehow induce you to click on a link in a document their server presents to you, the URL for which causes your phone to dial a telephone number. There should probably be some mechanism built in to examine the IP address of a server and only process phone number URLs from authorized servers.

6) Allow the user to edit a configuration file, and add entries to the systemwide configuration file, that cause a confirmatory pop-up to appear in response to certain dialing strings. The number would only be dialed if the user confirmed the request. This could be used to limit calls to local numbers, or selected area codes.

8. Technical Notes

1) The ISDN telephone only allows one active connection at a time. Among all "call appearances" (CAs), only one can be the "selected call appearance" (SCA, or the call actually connected to the phone's speaker and microphone); other CA buttons can only have either dialing or hold status.

2) To control/monitor the ISDN set, it may be placed in a special mode by issuing the following command sequence:

`at&&i`	(Initialize the ISDN phone)
`at&d3`	(Enable direct control/monitoring of ISDN set)
`at%a0=3`	(Set ISDN telephone current channel to voice)

This may be followed by commands to control/monitor particular ISDN functions: for example, the command "`at&&x0,4,1,1`" will will cause the computer to control and monitor the ISDN set's function buttons. In this state, you can no longer manually use the "Mute", "Speaker", "Conf"... buttons until the computer releases those controls. The disadvantage of using the control/monitor state in conjunction with an application is that if the application dies, the phone may be left in an indeterminate state.

3) The above direct-control mode allows the computer to mimic all possible human interactions with the ISDN set. However, communication must be synchronous, and hence the speed of interaction is very slow. Kelley Hu has developed a separate version of the Java application which operates in this mode. The application provided here does *not* put the ISDN telephone in direct-control mode, instead issuing pre-defined AT commands to achieve most common phone functions, like dial, conf, trans, hold, drop, etc.

4) To initiate a conference call, you can force CA2 to initiate the first call with the command:

`at*d/`<*phone number 1*>`/02`

but for the second call (third party call), specifying the CA will generate an error, so the second call should be initiated by the command:

`at*d/`<*phone number 2*>

5) In the case where you are away from your office and using *8510t* to place a call between yourself at your present location (*phone number 1*) and another party (*phone number 2*), it is desirable to automatically release the selected call appearance (SCA) on the ISDN telephone to allow the unit to be used to place another call, should someone be sitting by it. By experimentation, we discovered that this can be done by completing the commands required to establish both calls (which terminate with a second `at*k` command, see the code in *src/SerialConnection.java*) and then letting the java thread sleep for 2 seconds before issuing a `at*w02` command, which causes the call appearance which organized the conference call to be put on hold, releasing the SCA and suppressing the speaker.

6) Exit codes for the Java application are:

0: Conference call established

1: The phone connecting the computer is in use, or is in some other unknown state such that the AT command gets an error message

2: The number being called is busy

3: Java application caught an exception

7) According to the documentation accompanying the Java Communications API 2.0, *commapi/PlatformSpecific.html*, in the absence of root privileges you should be able to keep the file javax.comm.properties in the same directory as comm.jar. However, this did not appear to work for us. The search order for *javax.comm.properties* is:

   1) *<JDK>/lib*

   2) The directory that contains the first valid *comm.jar* that is included in the CLASSPATH environment variable.

   When we had the script point to */home/khu/download/commapi/comm.jar*, it worked, finding the proper *javax.comm.properties* file there. However, when the CLASSPATH environment variable pointed to:

   depot/package/8510t/classes/comm.jar

   the application failed by saying "port not found", that is, Java failed to find *javax.comm.properties* located in */depot/package/8510t/classes/*. It is therefore advisable to copy *javax.comm.properties* into */usr/java/lib*.

8) Perhaps the most difficult technical challenge posed by the Bourne shell script *8510t* was to figure out how to gracefully shut down remote processes that are triggered through the multihost dialing capability. when the parent shell process is interrupted or killed by the user. We thank Mr. Brian Katzung for suggesting a technically elegant solution. Examining the code in parallel with reading what follows will make the mechanism clearer. It relies on the creation of various signal handling functions, triggered by Bourne shell *trap* commands. Near the bottom of the *8510t* script, a do loop executes a Bourne shell NOP (:) while waiting on a read command. The script will in effect hang until it reads something from standard input There are three possible outcomes when this script is running on the local host:

   1) The main script receives a Control-C or KILL signal: the *mainsig* function is triggered by a trap, and kills any child Java processes; this in turn triggers the *java_watcher* function, which causes the script to drop out of the do loop, attempt to kill the Java process again (not needed for this outcome, but required for others), and exit.

   2) The Java dialing application succeeds and exits: the *java_watcher* function is triggered by a trap, and causes the script to exit.

   3) The Java dialing application fails and exits, and the *java_watcher* function is triggered by a trap. If the dialing failure was due to the phone being busy, and if multi-host dialing is allowed by the local configuration, *java_watcher* attempts to start *8510t* on a remote host; otherwise, the function causes the script to exit.

   If this script is running on a remote host in response to a multi-host dialing request, there are four possible outcomes (the first three above, plus a new fourth possibility):

   4) The parent script on the originating host receives a CONTROL-C or KILL signal: the read in the do loop receives something like an EOF, the script drops out of the loop, and kills any child Java process. This triggers the *java_watcher* function via a trap, and the function causes the script to exit.

   This mechanism could prove useful in many other Bourne shell scripts.

9. Acknowledgements

Brian Katzung provided helpful remarks about the *8510t* script. Ed Locke made contributions to an earlier version of the Java component of this application while working at LHNCBC as a summer worker in 1996.

10. References

1. *ISDN Voice Terminal 8510 Users Manual, Version 1.0*. NIH Office of Research Services.

2. AT&T Bell Laboratories. *Integrated Services Digital Network (ISDN) Application Programming Interface Programmer's Reference Manual*. AT&T, January 1993 (Dated July 1993).

3. *Integrated Services Digital Network (ISDN) ISDN 8510T Voice Terminal Feature Package 3 User's Manual*. AT&T.

4. *Integrated Services Digital Network (ISDN) 8500 Series Display Terminal Asynchronous Data Module User's Manual*. AT&T.

5. *ISDN 8510 Addendum, Addendum to Issue 2*. June 1993.

11. Appendix: UNIX Manual Page

Note that the file paths referred to in the following manual page are those used at the location at which *8510t* was written, and may not apply at your site.

Table of Contents